CERTIFICATION OF TRANSLATION

I, ***Moung-kyo Kim***, an employee of Y.P. LEE, MOCK & PARTNERS of Koryo Building, 1575-1 Seocho-dong, Seocho-gu, Seoul, Republic of Korea 137-875, hereby declare under penalty of perjury that I understand the Korean language and the English language; that I am fully capable of translating from Korean to English and vice versa; and that, to the best of my knowledge and belief, the statement in the English language in the attached translation of ***Korean Patent Application No. 10-2006-0070014*** consisting of 37 pages, have the same meanings as the statements in the Korean language in the original document, a copy of which I have examined.

Signed this 1st day of December 2006

Moung kyo Kim

FIG. 1

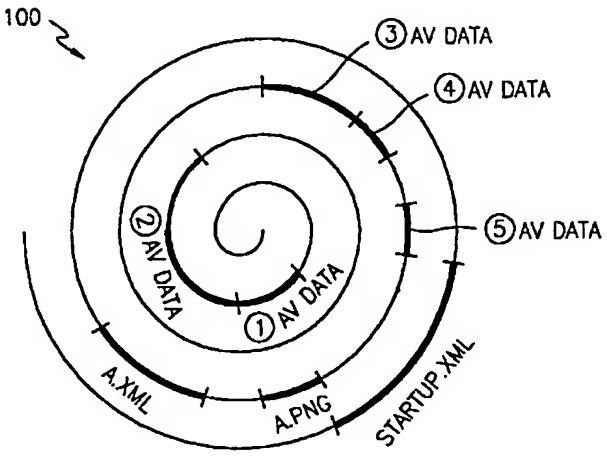


FIG. 2

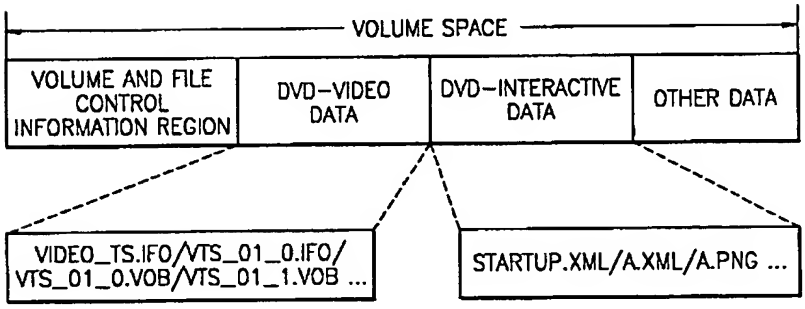


FIG. 3

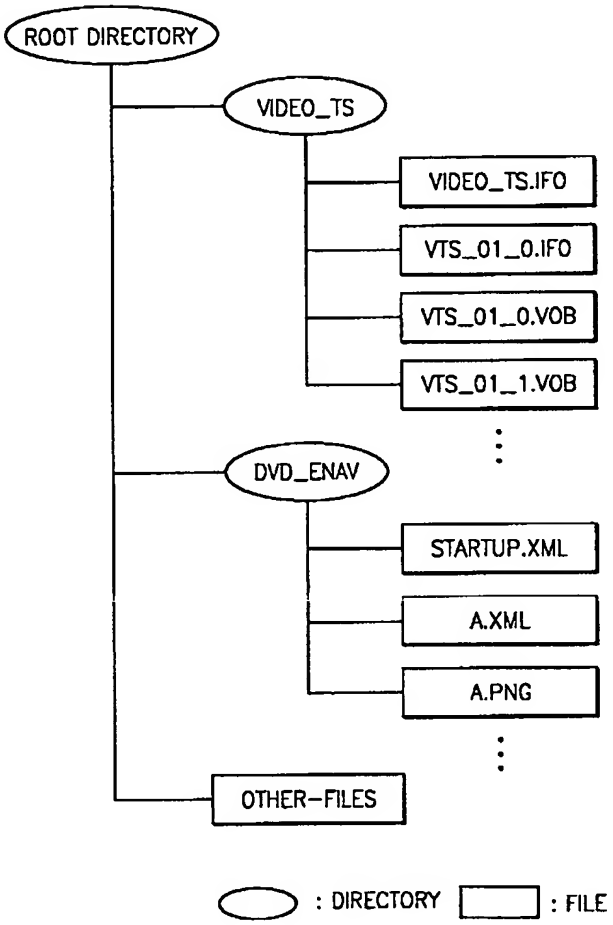

ROOT DIRECTORY
VIDEO_TS
VIDEO_TS.IFO
VTS_01_0.IFO
VTS_01_0.VOB
VTS_01_1.VOB
DVD_ENAV
STARTUP.XML
A.XML
A.PNG
OTHER-FILES
: DIRECTORY
: FILE

FIG. 4

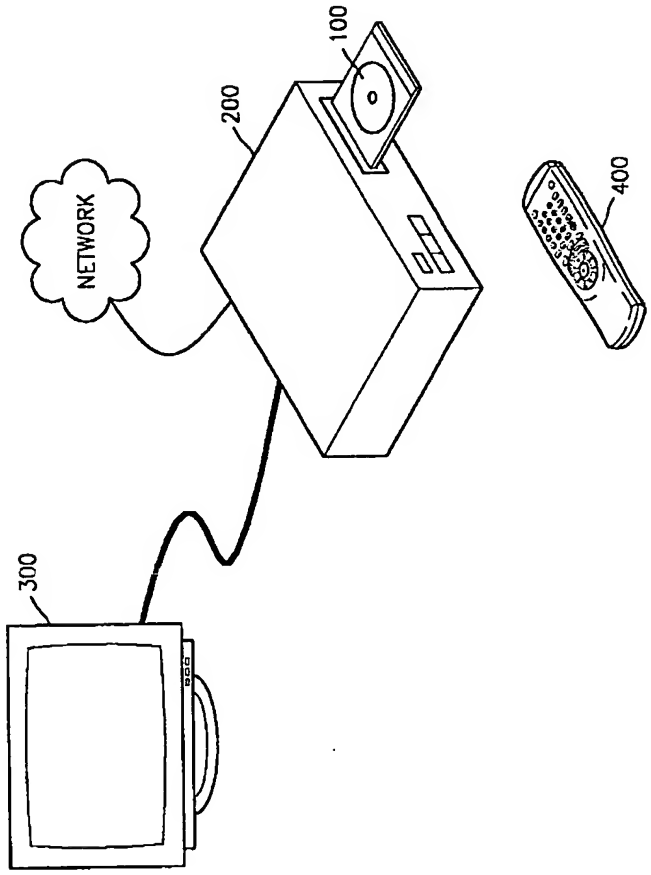


FIG. 5

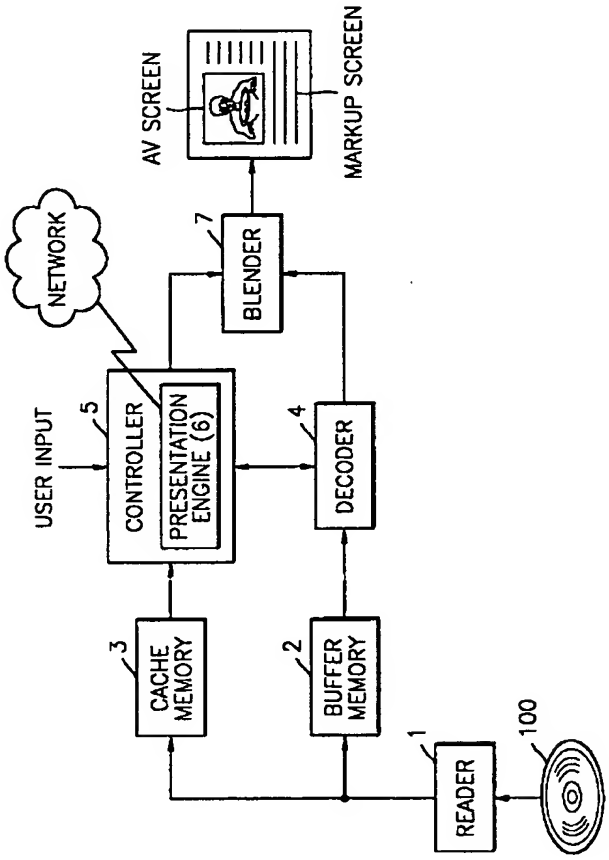

AV SCREEN
MARKUP SCREEN
NETWORK
BLENDER
7
USER INPUT
5
CONTROLLER
PRESENTATION ENGINE (6)
4
DECODER
3
CACHE MEMORY
2
BUFFER MEMORY
1
READER
100

FIG. 6

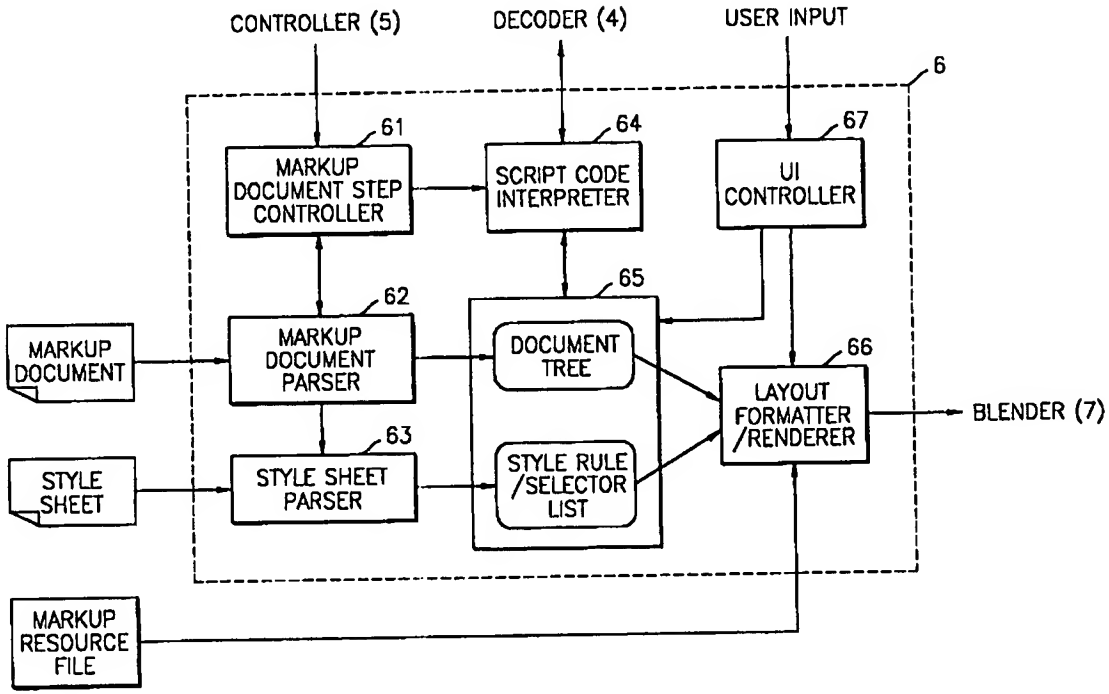

CONTROLLER (5)
DECODER (4)
USER INPUT
6
61
64
67
MARKUP DOCUMENT STEP CONTROLLER
SCRIPT CODE INTERPRETER
UI CONTROLLER
65
62
MARKUP DOCUMENT
MARKUP DOCUMENT PARSER
DOCUMENT TREE
66
LAYOUT FORMATTER /RENDERER
BLENDER (7)
63
STYLE SHEET
STYLE SHEET PARSER
STYLE RULE /SELECTOR LIST
MARKUP RESOURCE FILE

FIG. 7

```
<?xml version="1.0" ?>
<!—This is first comment -->
<!DOCTYPE html PUBLIC "-//DVD//DTD XHTML DVD-HTML 1.0//EN"
"http://www.dvdforum.org/enav/dtd/dvdhtml-1-0.dtd">
<html>
<head>
<title>DOM-core sample</title>
<script type="text/ecmascript">
alert("script code is in head tag")
function load_handler()
{
   alert("Hello, DVD-HTML")
   var strMsg="";
   strMsg+= document.documentElement.nodeName+"\n";
   strMsg+= document.documentElement.attributes.item(0).nodeName+"\n";
   var root= document.documentElement;
   for(var i=0;i<root.childNodes.length;i++)
   strMsg += "\t"+root.childNodes.item(i).nodeName+"\n";
   strMsg+= document.nodeName;
   alert(strMsg)
   alert(root.childNodes.item(0).childNodes.length);
alert(root.childNodes.item(0).childNodes.item(1).hasChildNodes());
}
function unload_handler()
{
   alert("good bye world")
}
</script>
</head>
<body onload="load_handler();" onunload="unload_handler();">
<script type="text/ecmacript">
alert(" before body text" )
</script>
<p>body text</p>
<script type="text/ecmascript">
<![CDATA[
alert("after body text")
]]>
</script>
</body>
</html>
```

FIG. 8

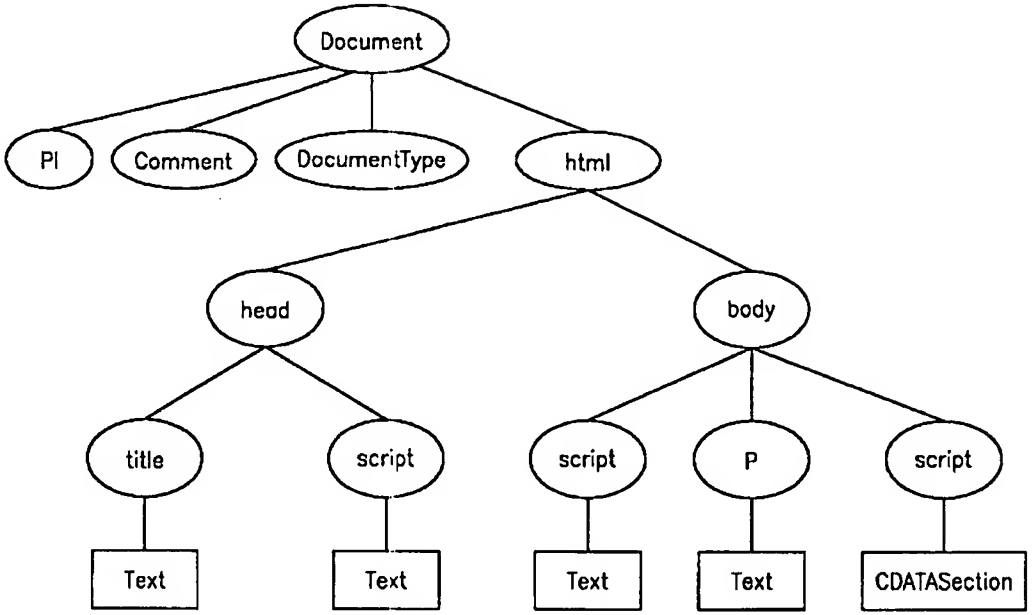

Document
PI
Comment
DocumentType
html
head
body
title
script
script
P
script
Text
Text
Text
Text
CDATASection

FIG. 9

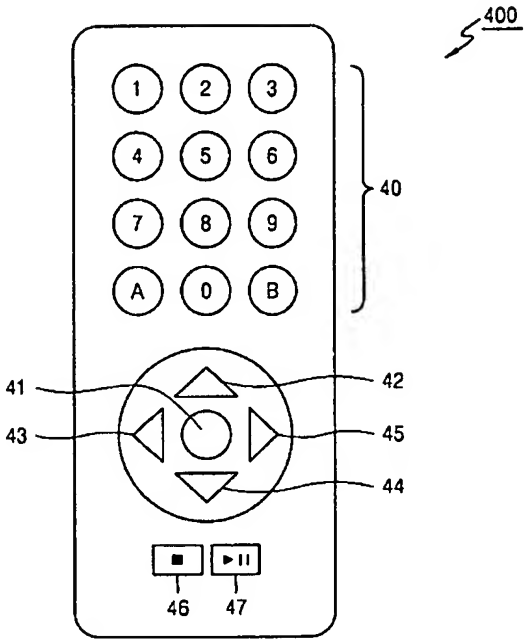

400
1
2
3
4
5
6
7
8
9
A
0
B
40
41
42
43
45
44
46
47

FIG. 10

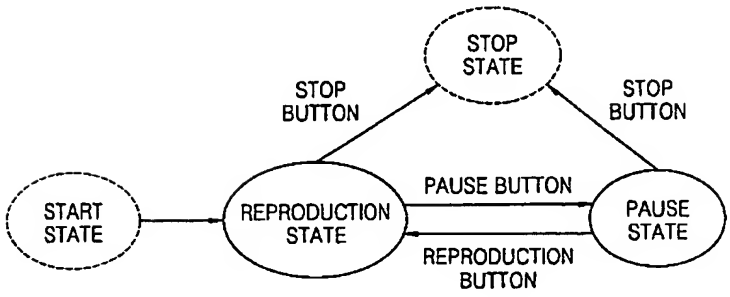

STOP STATE
STOP BUTTON
STOP BUTTON
START STATE
REPRODUCTION STATE
PAUSE BUTTON
PAUSE STATE
REPRODUCTION BUTTON

FIG. 11

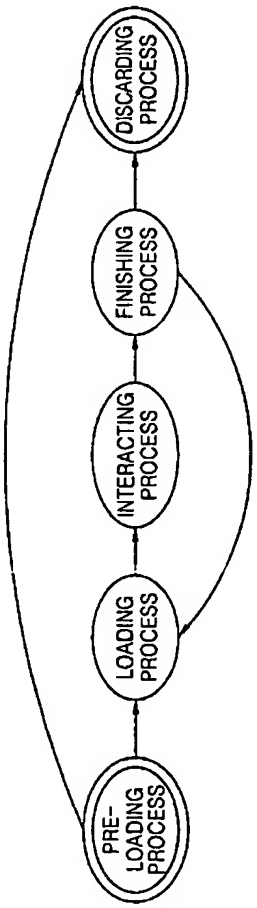

PRE-LOADING PROCESS
LOADING PROCESS
INTERACTING PROCESS
FINISHING PROCESS
DISCARDING PROCESS

FIG. 12B

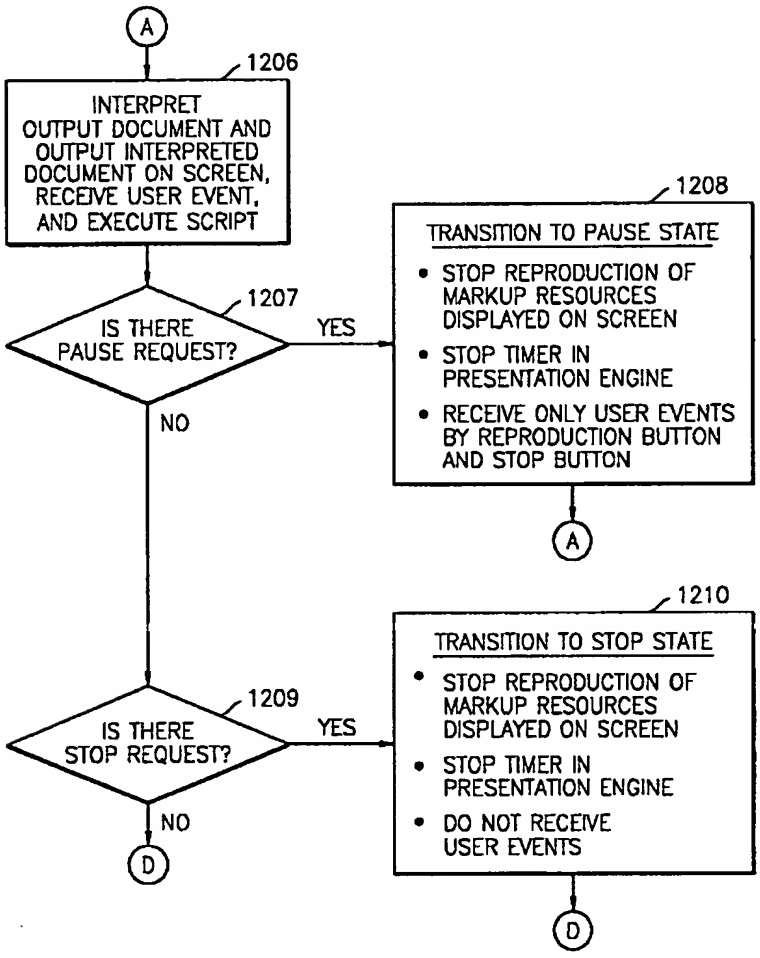

A
1206
INTERPRET OUTPUT DOCUMENT AND OUTPUT INTERPRETED DOCUMENT ON SCREEN, RECEIVE USER EVENT, AND EXECUTE SCRIPT
1207
IS THERE PAUSE REQUEST?
YES
NO
1208
TRANSITION TO PAUSE STATE
• STOP REPRODUCTION OF MARKUP RESOURCES DISPLAYED ON SCREEN
• STOP TIMER IN PRESENTATION ENGINE
• RECEIVE ONLY USER EVENTS BY REPRODUCTION BUTTON AND STOP BUTTON
A
1209
IS THERE STOP REQUEST?
YES
NO
1210
TRANSITION TO STOP STATE
• STOP REPRODUCTION OF MARKUP RESOURCES DISPLAYED ON SCREEN
• STOP TIMER IN PRESENTATION ENGINE
• DO NOT RECEIVE USER EVENTS
D
D

FIG. 12C

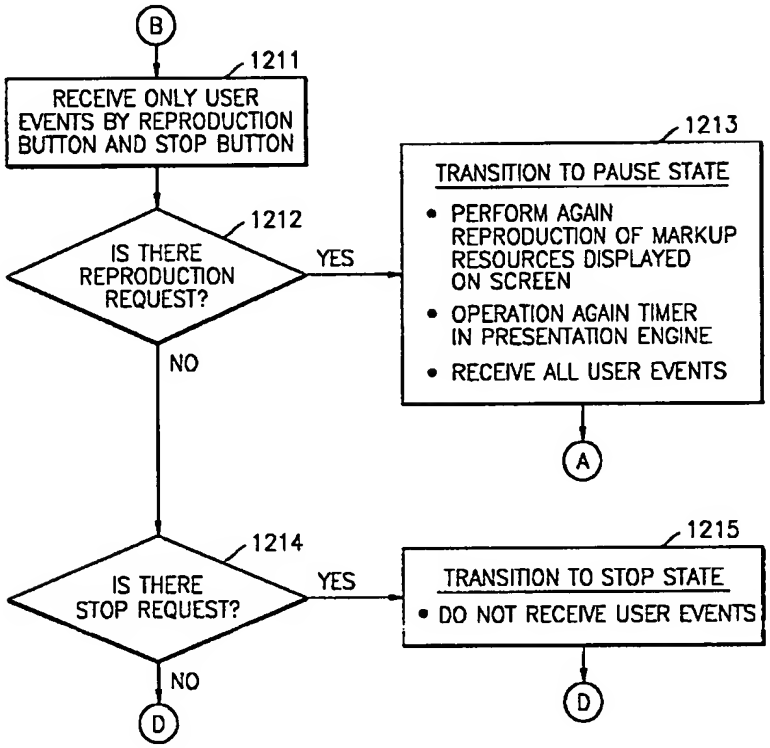

B
1211
RECEIVE ONLY USER EVENTS BY REPRODUCTION BUTTON AND STOP BUTTON
1212
IS THERE REPRODUCTION REQUEST?
YES
NO
1213
TRANSITION TO PAUSE STATE
• PERFORM AGAIN REPRODUCTION OF MARKUP RESOURCES DISPLAYED ON SCREEN
• OPERATION AGAIN TIMER IN PRESENTATION ENGINE
• RECEIVE ALL USER EVENTS
A
1214
IS THERE STOP REQUEST?
YES
NO
1215
TRANSITION TO STOP STATE
• DO NOT RECEIVE USER EVENTS
D
D

FIG. 12D

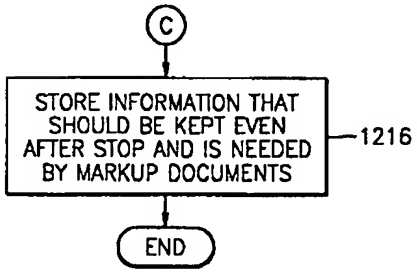
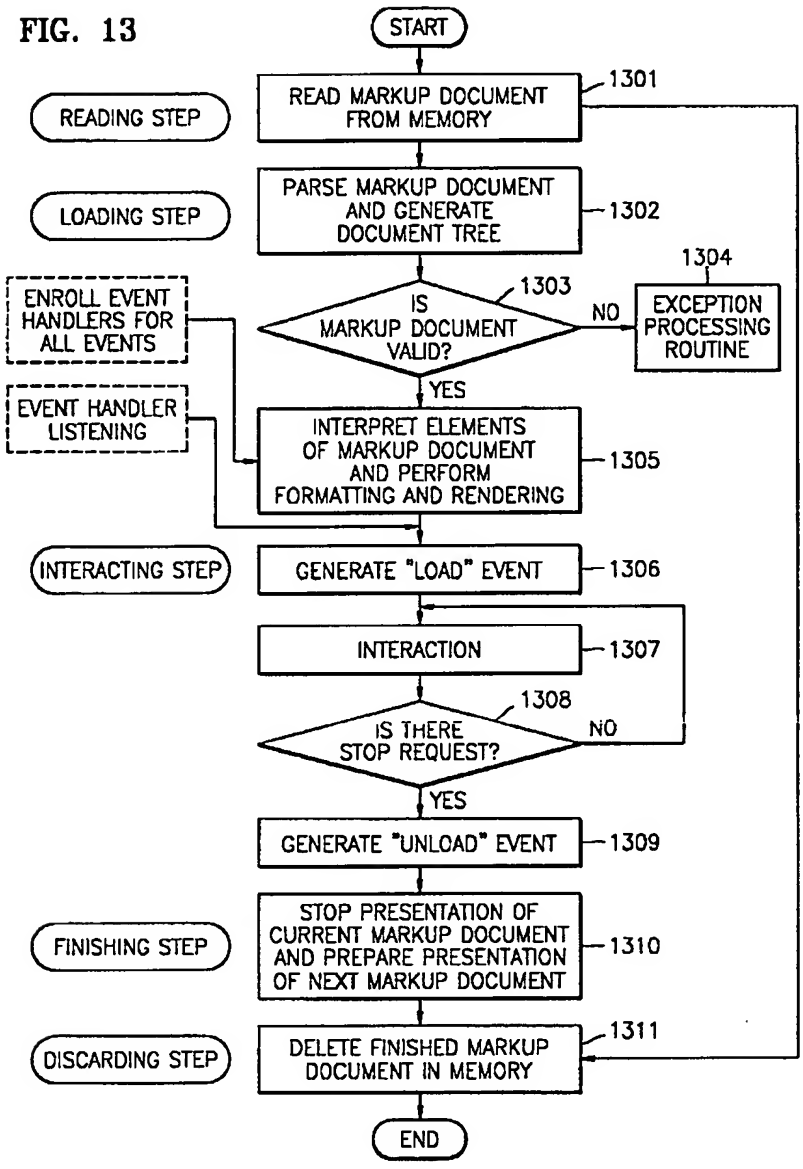

C
STORE INFORMATION THAT SHOULD BE KEPT EVEN AFTER STOP AND IS NEEDED BY MARKUP DOCUMENTS
1216
END

FIG. 13


START
READING STEP
READ MARKUP DOCUMENT FROM MEMORY
1301
LOADING STEP
PARSE MARKUP DOCUMENT AND GENERATE DOCUMENT TREE
1302
1304
ENROLL EVENT HANDLERS FOR ALL EVENTS
IS MARKUP DOCUMENT VALID?
1303
NO
EXCEPTION PROCESSING ROUTINE
YES
EVENT HANDLER LISTENING
INTERPRET ELEMENTS OF MARKUP DOCUMENT AND PERFORM FORMATTING AND RENDERING
1305
INTERACTING STEP
GENERATE "LOAD" EVENT
1306
INTERACTION
1307
IS THERE STOP REQUEST?
1308
NO
YES
GENERATE "UNLOAD" EVENT
1309
FINISHING STEP
STOP PRESENTATION OF CURRENT MARKUP DOCUMENT AND PREPARE PRESENTATION OF NEXT MARKUP DOCUMENT
1310
DISCARDING STEP
DELETE FINISHED MARKUP DOCUMENT IN MEMORY
1311
END

ABSTRACT

[Abstract of the Disclosure]

A reproducing method and apparatus for interactive mode using markup documents are provided. The method for reproducing AV data in interactive mode comprises a presentation engine operating according predefined states, wherein the operating state of a presentation engine for reproducing a markup document is divided into and defined as a start state, a reproduction state, a pause state, and stop state. In the reproduction state, the presentation engine performs a loading step for interpreting a markup document and loading the markup document on a screen; an interacting step for performing interaction between the markup document loaded on the screen with a user; and a finishing step for finishing the markup document loaded on the screen. By the method, when AV data are reproduced in the interactive mode, compatibility of display is provided.

[Representative Drawing]

FIG. 9

SPECIFICATION

[Title of the Invention]

REPRODUCING APPARATUS FOR INTERACTIVE MODE USING MARKUP DOCUMENTS

[Brief Description of the Drawings]

FIG. 1 is a schematic diagram of an interactive DVD on which AV data is recorded;

FIG. 2 is a schematic diagram of a volume space in the interactive DVD of FIG. 1;

FIG. 3 is a diagram showing the directory structure of an interactive DVD;

FIG. 4 is a schematic diagram of a reproducing system according to a preferred embodiment of the present invention;

FIG. 5 is a functional block diagram of a reproducing apparatus according to a preferred embodiment of the present invention;

FIG. 6 is a diagram of an example of the presentation engine of FIG. 5;

FIG. 7 is a diagram showing an example of a markup document;

FIG. 8 is a diagram of a document tree generated based on the markup document of FIG. 7;

FIG. 9 is a diagram of an example of a remote controller;

FIG. 10 is a state diagram showing each state of a presentation engine and the relations between the states. The states and relations between the states are defined to reproduce a markup document;

FIG. 11 is a diagram showing a document life cycle in a reproduction state of FIG. 10;

FIGS. 12a through 12d are a flowchart of the steps performed by a reproducing method according to a preferred embodiment of the present invention; and

FIG. 13 is a flowchart of the steps performed by a reproducing method according to another preferred embodiment of the present invention.

[Detailed Description of the Invention]

[Object of the Invention]

[Technical Field of the Invention and Related Art prior to the Invention]

The present invention relates to reproduction of markup documents, and more particularly, to a method and apparatus for reproducing audio/visual (AV) data in interactive mode using markup documents.

Interactive digital versatile discs (DVD), from which data can be reproduced in interactive mode by loading them in a DVD drive installed in a personal computer (PC), are being sold in the marketplace. An interactive DVD is a DVD on which markup documents are recorded together with AV data. AV data recorded on the interactive DVD can be reproduced in two ways. One is video mode in which data is displayed as a normal DVD, and the other is interactive mode in which reproduced AV data is displayed through a display window defined by a markup language document. If the interactive mode is selected by a user, a browser in the PC interprets and displays a markup language document recorded on the interactive DVD. AV data selected by the user is displayed in the shown display window of the markup language document. A leading markup language document is an extensible markup language (XML) document.

For example, when AV data is a movie, moving pictures are output on the display window of the XML document, and a variety of additional information such as the script and synopsis of the movie, and photos of actors is displayed on the remaining part of the screen. The additional information includes image files or text files. In addition, the displayed markup document enables interaction. For example, if the user pushes a button prepared on the markup document, then a brief personal description of an actor in the moving picture being reproduced at present is displayed.

A browser is used as a markup document viewer that can interpret and display markup documents recorded on an interactive DVD. Leading browsers include Microsoft Explorer and Netscape Navigator. However, since these browsers have different processes for interpreting and displaying markup documents, when an identical interactive DVD is reproduced in interactive mode, displays by these browsers may be different to each other. That is, display compatibility between theses browsers is not provided. Also, while a browser performs a process for reproducing a markup document (a process for interpreting and displaying the markup document), the user cannot pause the operation.

[Technical Goal of the Invention]

The present invention provides a method and apparatus which can control a process of reproducing markup documents when AV data is reproduced in interactive mode using the markup documents.

The present invention also provides a method and apparatus which interpret and display markup documents when AV data is reproduced in interactive mode using the markup documents, such that display compatibility is provided.

[Structure and Operation of the Invention]

According to an aspect of the present invention, there is provided a method for reproducing audio/visual data, including audio data and/or video, in interactive mode, the method comprising: interpreting a markup document and loading the markup document on a screen; a user performing interaction with the markup document loaded on the screen; and finishing the markup document loaded on the screen.

Before the loading step, the method may further comprise reading and fetching the markup document to a memory. After the finishing step, the method may further comprise deleting the markup document in the memory.

In the method, the loading step may comprise (a) interpreting the markup document and generating a document tree; and (c) rendering the markup document

based on the generated document tree. In the method, the reading step may further comprise reading and fetching a stylesheet for the markup document to the memory.

In the method, the loading step may comprise (a) interpreting the markup document and generating a document tree; (b) interpreting the stylesheet and applying the stylesheet to the document tree; (c1) based on the document tree to which the stylesheet has been applied, generating a formatting structure; and (c2) based on the generated formatting structure, rendering the markup document.

In the step (a) of the method, the document tree may be generated according to a rule that a root node of all nodes is set to a document node, a rule that all texts and elements generate nodes, and a rule that a processing instruction, a comment, and a document type generate a node.

According to another aspect of the present invention, there is provided an apparatus for reproducing in interactive mode AV data including audio data and/or video data recorded on an information storage medium, the apparatus comprising: a reader which reads and fetches data recorded on the information storage medium; a cache memory which temporarily stores a markup document that is read by the reader; and a presentation engine which presents the markup document according to a document life cycle which comprises a loading step for interpreting the markup document read by the reader and loading the document on a screen, an interacting step for performing interaction between the markup document loaded on the screen and the user, and a finishing step for finishing the presentation of the markup document.

In the apparatus, before the loading step the presentation engine may perform a reading step for reading and fetching the markup document to the cache memory, as part of the document life cycle. In the apparatus, after the finishing step the presentation engine may perform a discarding step for deleting the markup document remaining in the cache memory, as part of the document life cycle.

In the apparatus, in the loading step, the presentation engine may perform steps (a) interpreting the markup document and generating a document tree; and (c) based on the generated document tree, rendering the markup document.

In the apparatus, the presentation engine may further read and fetch a stylesheet for the markup document from the memory, and perform as the loading step: (a) interpreting the markup document and generating a document tree; (b) interpreting the stylesheet and applying the stylesheet to the document tree; (c1) based on the document tree to which the stylesheet has been applied, generating a formatting structure; and (c2) based on the generated formatting structure, rendering the markup document.

In the apparatus, the presentation engine may generate the document tree according to rules that a root node of all nodes is set to a document node, all texts and elements generate nodes, and a processing instruction, a comment, and a document type generate a node.

According to still another aspect of the present invention, there is provided an apparatus for reproducing AV data including audio data and/or video data recorded on an information storage medium, in interactive mode, the apparatus comprising: a reader which reads and fetches data recorded on the information storage medium; a cache memory which temporarily stores a markup document and a stylesheet that are read by the reader; and a presentation engine which comprises a markup document parser which interprets the markup document and generates a document tree, a stylesheet parser which interprets the stylesheet and generates a style rule/selector list, a script code interpreter which interprets a script code contained in the markup document, a document object model (DOM) logic unit which modifies the document tree and the style rule/selector list according to interaction with the script code interpreter, and a layout formatter/renderer which applies the document tree and stylesheet rule/selector list to the document tree, based on the applying, generates a formatting structure, and based on the generated formatting structure, renders the markup document.

In the apparatus, the markup document parser may generate the document tree according to rules that a root node of all nodes is set to a document node, all texts and elements generate nodes, and a processing instruction, a comment, and a document type generate a node.

In the apparatus, the presentation engine may comprise a markup document step controller, and the markup document step controller may generate a 'load' event to the script code interpreter if the rendering of the markup document is completed. The step controller may generate an 'unload' event to the script code interpreter in order to finish presentation of the markup document.

Preferred embodiments of the present invention will now be described with reference to the attached drawings.

Referring to FIG. 1, in the tracks of an interactive DVD 100, AV data are recorded as MPEG bitstreams and a plurality of markup documents are recorded. Here, the markup documents indicate any documents, to which source codes that are written in Script language or Java language are linked or inserted, as well as those documents that are written in markup languages such as hyper text markup language (HTML) and XML. In other words, the markup documents play a role of a kind of application that is needed when AV data is reproduced in the interactive mode. Meanwhile, image files, animation files, and sound files that are linked to and embedded into a markup document and are reproduced are referred to as 'markup resources'.

FIG. 2 is a schematic diagram of a volume space in the interactive DVD 100 of FIG. 1.

Referring to FIG. 2, the volume space of the interactive DVD 100 comprises a control information region in which volume and file control information is recorded, a DVD-Video data region in which video title data corresponding to the control information are recorded, and a DVD-Interactive data region in which data that are needed in order to reproduce AV data in interactive mode are recorded.

In the DVD-Video data region, VIDEO_TS.IFO that has reproduction control information of all the included video titles and VTS_01_0.IFO that has reproduction control information of a first video title are first recorded and then VTS_01_0.VOB, VTS_01_1.VOB, ..., which are AV data forming video titles, are recorded. VTS_01_0.VOB, VTS_01_1.VOB, ..., are video titles, that is, video objects (VOBs). Each VOB contains VOBUs in which navigation packs, video packs, and audio packs

are packed. The structure is disclosed in more detail in a draft standard for DVD-Video, "DVD-Video for Read Only Memory Disc 1.0".

DVD_ENAV.IFO, which has reproduction control information of all interactive information, a start document STARTUP.XML, a markup document file A.XML, and a graphic file A.PNG, which is a markup resource to be inserted into A.XML and displayed, are recorded in the DVD-Interactive data region. Other markup documents and markup resource files having a variety of formats that are inserted into the markup documents may also be recorded.

FIG. 3 is a diagram showing the directory structure of the interactive DVD 100.

Referring to FIG. 3, a DVD video directory VIDEO_TS and a DVD interactive directory DVD_ENAV in which interactive data are recorded are prepared in the root directory.

VIDEO_TS.IFO, VTS_01_0.IFO, VTS_01_0.VOB, VTS_01_1.VOB, ..., which are explained referring to FIG. 2, are stored in the VIDEO_TS. STARTUP.XML, A.XML, and A.PNG, which are explained referring to FIG. 2, are stored in the DVD_ENAV,.

FIG. 4 is a schematic diagram of a reproducing system according to a preferred embodiment of the present invention.

Referring to FIG. 4, the reproducing system comprises an interactive DVD 100, a reproducing apparatus 200, a TV 300, which is a display apparatus according to the present embodiment, and a remote controller 400. The remote controller 400 receives a control command from the user and transmits the command to the reproducing apparatus 200. The reproducing apparatus 200 has a DVD drive which reads data recorded on the interactive DVD 100. If the DVD 100 is placed in the DVD drive and the user selects the interactive mode, then the reproducing apparatus reproduces desired AV data in the interactive mode by using a markup document corresponding to the interactive mode, and sends the reproduced AV data to the TV 300. AV scenes of the reproduced AV data and a markup scene from the markup document are displayed together on the TV 300. The "interactive mode" is a reproducing mode in which AV data are displayed as AV scenes in a display window defined by a markup document,

that is, a reproducing mode in which AV scenes are embedded in a markup scene and then displayed. Here, the AV scenes are scenes that are displayed on the display apparatus when the AV data are reproduced, and the markup scene is a scene that is displayed on the display apparatus when the markup document is parsed. Meanwhile, the "video mode" indicates a prior art DVD-Video reproducing method, by which only AV scenes that are obtained by reproducing the AV data are displayed. In the present embodiment, the reproducing apparatus 200 supports both the interactive mode and video mode. In addition, the reproducing apparatus can transmit or receive data after being connected to a network, such as the Internet.

FIG. 5 is a functional block diagram of the reproducing apparatus 200 according to a preferred embodiment of the present invention.

Referring to FIG. 5, the reproducing apparatus 200 comprises a reader 1, a buffer memory 2, a cache memory 3, a controller 5, a decoder 4, and a blender 7. A presentation engine 6 is included in the controller 5. The reader 1 has an optical pickup (not shown) which reads data by shining a laser beam on the DVD 100.

The reader 1 controls the optical pickup according to a control signal from the controller 5 such that the reader reads AV data and markup documents from the DVD 100.

The buffer memory 2 buffers AV data. The cache memory 3 is used for temporarily storing a reproduction control information file for controlling reproduction of AV data and/or markup documents recorded on the DVD 100, or other needed information.

In response to a user's selection, the controller 5 controls the reader 1, the presentation engine 6, the decoder 4, and the blender 7 so that the AV data recorded on the DVD 100 are reproduced in the video mode or interactive mode.

The presentation engine 6 which is part of the controller 5 is an interpretation engine which interprets and executes markup languages and client interpretation program languages, for example, JavaScript and Java. In addition, the presentation engine 6 may further include a variety of plug-in functions. The plug-in function

enables markup resource files to be opened with a variety of formats, which are included in or linked to a markup document. That is, the presentation engine 6 plays a role of a markup document viewer. Also, in the present embodiment, the presentation engine 6 can be connected to the Internet and read and fetch predetermined data.

In the interactive mode, the presentation engine 6 fetches a markup document stored in the cache memory 3, interprets the document and performs rendering. The blender 7 blends an AV data stream and the rendered markup document such that the AV data stream is displayed in a display window defined by the markup document, i.e., the AV scene is embedded in the markup scene. Then, the blender 7 outputs the blended scene to the TV 300.

In a process for reproducing (that is, interpreting and displaying) a markup document according to the present invention, the presentation engine 6 defines 1) a start state in which operations for start of reproduction are performed, 2) a reproduction state in which a markup document is executed, 3) a pause state in which the reproduction of the markup document is temporarily stopped, and 4) a stop state in which the reproduction of the markup document is stopped, and operates based on the defined states. The '1) start state' indicates a state in which the presentation engine 6 performs operations for initialization. The operations of the presentation engine 6 in the '2) reproduction state', '3) pause state', and '4) stop state' are determined by a user event that is generated by the remote controller 400 according to a user input, and a script code that is written in the markup document. This will be explained later in more detail.

In addition, according to the present invention, the presentation engine 6 presents a markup document in the reproduction state, based on a document life cycle which comprises a reading step where the markup document is read from the cache memory 3, a loading step where the markup document read by the reader 1 is interpreted and loaded on the screen, an interacting step where interaction between the markup document loaded on the screen and the user is performed, a finishing step where the markup document loaded on the screen is finished, and a discarding step

where the markup document remaining in the cache memory 3 is deleted.

FIG. 6 is a diagram of an example of the presentation engine of FIG. 5.

Referring to FIG. 6, the presentation engine 6 comprises a markup document step controller 61, a markup document parser 62, a stylesheet parser 63, a script code interpreter 64, a document object model (DOM) logic unit 65, a layout formatter/renderer 66, and a user interface (UI) controller 67.

The markup document parser 62 interprets a markup document and generates a document tree. The rules for generating a document tree are as follows. First, a root node of all nodes is set as a document node. Secondly, all texts and elements generate nodes. Thirdly, a processing instruction, a comment, and a document type generate a node. FIG. 7 is a diagram showing an example of a markup document. FIG. 8 is a diagram of a document tree generated based on the markup document of FIG. 7. Thus, according to the present invention, an identical document tree is generated for an identical markup document.

The UI controller 67 receives a user input through the remote controller 400, and sends it to the DOM logic unit 65 and/or the layout formatter/renderer 66. That is, the UI controller 67 generates a user event according to the present invention.

The stylesheet parser 63 parses a stylesheet and generates a style rule/selector list. The stylesheet enables the form of a markup document to be freely set. In the present embodiment, the syntax and form of a stylesheet comply with the cascading style sheet (CSS) processing model of the World Wide Web Consortium (W3C). The script code interpreter 64 interprets a script code included in the markup document. With the DOM logic unit 65, the markup document can be made into a program object or can be modified. That is, the document tree and the style rule/selector list are modified or improved according to the interaction with the script code interpreter 64, or a user event from the UI controller 67. The layout formatter/renderer 66 applies the style rule/selector list to a document tree, and according to a document form (for example, whether the form is a printed page or sound) that is output based on the applying, generates a formatting structure corresponding to the form, or changes a formatting

structure according to a user event from the UI controller 67. Though the formatting structure looks like a document tree at first glance, the formatting structure can use a pseudo-element and does not necessarily have a tree structure. That is, the formatting structure is dependent on implementation. Also, the formatting structure may have more information than a document tree has or may have less information. For example, if an element of a document tree has a value "none" as an attribute value of "display", the element does not generate any value for a formatting structure. Since the formatting structure of the present embodiment complies with a CSS2 processing model, more detailed explanation is available at the CSS2 processing model. The layout formatter/renderer 66 renders a markup document according to the form of a document (that is, a target medium) that is output based on the generated formatting structure, and outputs the result to the blender 7. For the rendering, the layout formatter/renderer 66 may have a decoder for interpreting and outputting an image or sound. In this manner, the layout formatter/renderer 66 decodes a markup resource linked to the markup document and outputs the markup resource to the blender 7.

The markup document step controller 61 controls steps so that interpretation of a markup document is performed according to the document life cycle described above. Also, if the rendering of a markup document is finished, the markup document step controller 61 generates a 'load' even to the script code interpreter 64, and in order to finish presentation of a markup document, generates an 'unload' event to the script code interpreter 64.

FIG. 11 is a diagram of an example of a remote controller.

Referring to FIG. 11, a group of numerical buttons and special character buttons 40 is arranged at the top of the front surface of the remote controller 400. At the center of the front surface, a direction key 42 for moving upward a pointer displayed on the screen of the TV 300, a direction key 44 for moving the pointer downward, a direction key 43 for moving the pointer to the left, and a direction key 45 for moving the pointer to the right are arranged, and an enter key 41 is arranged at the center of the direction keys.

At the bottom of the front surface, a stop button 46 and a reproduction/pause button 47 are arranged. The reproduction/pause button 47 is prepared as a toggle type such that whenever the user pushes the button 48, the reproduction function and pause function are selected alternately. According to the present invention, the user can control the reproduction process of a markup document by the presentation engine 6, by pushing the stop button 46 and reproduction/pause button 47 in the interactive mode.

FIG. 10 is a state diagram showing each state of the presentation engine 6 and the relations between the states, the states and relations that are defined to reproduce a markup document.

Referring to FIG. 10, the states of the presentation engine 6 are broken down into 1) a start state, 2) a reproduction state, 3) a pause state, and 4) stop state. 1) In the start state, if there is a DVD 100 in the reproducing apparatus 200, the presentation engine 6 performs initialization operations such as reading and fetching disc information, or loading a file system to the cache memory 3. The initialization state is achieved inside the reproducing apparatus and is not recognized by the user. If the initialization operations are completed, the state of the presentation engine 6 is transited to the reproduction state. 2) In the reproduction state, the presentation engine 6 reproduces a markup document that is specified as a start document. If the user pushes the pause button 48 on the remote controller 400, the state of the presentation engine 6 is transited to the pause state. 3) Pause of reproduction of a markup document means pause of reproduction of markup resources that are linked to the markup document and displayed on the markup scene. For example, in a case where a flash animation is embedded in the markup scene and is being displayed, the motion of the flash animation stops during the pause state. If the user pushes the reproduction/pause button 48 again, the state of the presentation engine 6 is transited to the reproduction state and the reproduction of the markup document begins again. That is, the reproduction of the markup resources displayed on the markup scene begins again from the part where the markup resources stopped. The state of the presentation engine 6

alternates between the reproduction state and the pause state when the reproduction/pause button 48 is pushed. Meanwhile, if the user pushes the stop button 47 in the pause state or the reproduction state, the state of the presentation engine 6 is transited to the stop state where the reproduction of the markup document stops completely. 4) In the stop state, the reproduction of markup resources displayed on the markup stops completely. Accordingly, if the user pushes the reproduction/pause button 48 again, reproduction begins again from the first part of the markup resources.

The operations of the presentation engine 6 in the 1) start state, 2) reproduction state, 3) pause state, and 4) stop state are determined by user events that are generated by the remote controller 400 according to a user input, and script codes written in the markup document. Accordingly, by changing the user events and script codes written in the markup document, the operations of the presentation engine 6 in respective states can be changed in a variety of ways.

FIG. 10 is a diagram showing a document life cycle in a reproduction state of FIG. 10.

Referring to FIG. 10, the document life cycle comprises a reading step, a loading step, an interacting step, a finishing step, and a discarding step. All markup documents go through the document life cycle according to the present invention. However, some markup documents may go through a document life cycle in which the discarding step immediately follows the reading step. A case where a markup document is stored in the cache memory 3 and then deleted without being presented (displayed) corresponds to this cycle. Also, there may be a document life cycle in which the loading step is performed again after the finishing step. A case where a markup document whose presentation has finished is being presented again corresponds to this cycle.

The reading step ends in a process in which a markup document (and a stylesheet) is read by the cache memory 3. That is, a resource related to the markup document is generated as an on-memory item.

The loading step includes processes for interpreting the markup document and

presenting the markup document on the display screen. That is, the "loading" in the loading step means that the markup document is loaded on the screen. The interpreting of the markup document indicates a process for performing a syntax check for checking whether or not the syntax of a code is correct and a document type definition (DTD) check for checking whether or not there is a semantic error, and if there is no error, generating a document tree. Also, the interpreting includes a process for interpreting a stylesheet which exists separately from the markup document or is included in the markup document.

For an XML document, the syntax checking process includes checking whether or not XML elements are properly arranged. That is, it is checked whether or not tags that are XML elements are tested in accordance with the syntax. A detailed explanation of the syntax check is available in the XML standard. The DTD is information on document rules accompanying a markup document and distinguishes tags of the document, identifies attribute information set to tags, and indicates how values appropriate to the attribute information are set. In the DTD checking process, a semantic error of the markup document is found based on the DTD. The rules that are applied to a process for generating a document tree according to the present invention are the same as described above.

In brief, the loading step includes the process for interpreting the markup document and generating a document tree, and the process for rendering the markup document based on the generated document tree. More specifically, in the loading step, a document tree is generated by interpreting the markup document, a style rule/selector list is generated by interpreting the stylesheet, the generated style rule/selector list is applied to the document tree, a formatting structure is generated based on the type of list applied, and the markup document is rendered based on the formatting structure.

In the interacting step, the displayed content of a document changes, for example, by an interaction with the user when the user pushes a button of a document loaded on the screen or scrolls the screen, or by an interaction between the decoder 4

and the presentation engine 6, or by a process in which the user pushes a button on the remote controller 400 to control the reproduction of the markup document. In the interacting step, the markup document presented on the screen receives a load event from the markup document step controller 61. If the screen displays another markup document shifting away from the currently loaded markup document, an unload event is generated. If the user pushes a button on the remote controller 400, a user input event is sent to the script code interpreter 64 through the UI controller 67 and the DOM controller 65. At this time, it is determined whether or not to reflect an event in the presentation engine 6 after an event handler script code that is provided to the DOM controller 65 is executed in the script code interpreter 64. Then, if it is determined to reflect the event in the presentation engine 6, the event is reflected and processed in the presentation engine 6 to perform a predefined operation. For example, when any one of the reproduction/pause button 47 and the stop button 46 that control the execution states of the reproducing apparatus is pushed, the operation for navigating elements forming the markup documents such as the direction keys 42 through 45 and the enter key 41 corresponds to this. If the user does not want to reflect the event, the user can use a function, event.preventDefault( ), which is provided by the WC3. Detailed information is described in Document Object Model (DOM) Level 2 Events Specification version 1.0.

The finishing step indicates a state where the presentation of a markup document is finished and the markup document remains in the cache memory 3.

In the discarding step, the markup document whose presentation is finished is deleted from the cache memory 3. That is, in the discarding step, the on-memory item information is deleted.

Based on the structure described above, a reproduction method according to the present invention will now be explained.

FIGS. 12a through 12d are a flowchart of the steps performed by a reproducing method according to a preferred embodiment of the present invention.

Referring to FIG. 12a, if there is a DVD 100 in the reproducing apparatus 200,

the reproducing apparatus initializes the presentation engine 6 in step 1201, and sets STARTUP.XML as an output document in step 1202. Based on the user input event that is generated when a user input button is pushed, the presentation engine 6 determines the current state. If the current state is a reproduction state in step 1203, A is performed, if it is a pause state in step 1204, B is performed, and if it is a stop state in step 1205, C is performed.

Referring to FIG. 12b, if the current state is a reproduction state (A), the presentation engine 6 interprets and displays on the screen STARTUP.XML, which is set to the output document, receives a user event from the user input, and executes a script corresponding to the user event, the script which is written in or linked to the markup document in step 1206. If there is a pause request from the user, that is, if the user pushes the pause button 48 in step 1207, the state is transited to the pause state in step 1208. In the pause state, the reproduction of markup resources that are displayed on the screen stops, and a timer which is needed in interpreting markup documents and in decoding markup resources in the presentation engine 6 stops. In the pause state, only user events corresponding to the reproduction button 48 and stop button 47 are received. Even if any of the other buttons, for example, the pause button, is pushed, the presentation engine 6 does not perform an operation corresponding to the button. If there is a stop request from the user, that is, if the user pushes the stop button 47 in step 1209, the state is transited to the stop state in step 1210. In the stop state, the presentation engine 6 completely stops the reproduction of markup resources that are displayed on the screen, completely stops the timer, and does not receive any user events.

Referring to FIG. 12c, in the pause state (B), if the user pushes the reproduction button 48 or the stop button 47, the presentation engine 6 receives a user event corresponding to the button in step 1211. That is, if there is a reproduction stop request from the user, that is, if the user pushes the stop button 48 in step 1212, the state is transited to the reproduction state in step 1213. In the reproduction state, the presentation engine 6 begins reproduction of the markup resources displayed on the

screen from a part where the reproduction stopped temporarily, begins the timer from a part where the timer stopped, and receives all user events. If there is a reproduction stop request from the user, that is, if the user pushes the stop button 46 in step 1214, the state is transited to the stop state in step 1215. In the stop state, the presentation engine 6 does not receive any user events.

Referring to FIG. 12d, in the stop state (C), the presentation engine 6 stores information that should be kept even after the stop and is needed by markup documents, in a non-volatile memory (not shown) in step 1216.

FIG. 13 is a flowchart of the steps performed by a reproducing method according to another preferred embodiment of the present invention.

FIG. 13 shows processes for processing a markup document in each state of the document life cycle. That is, in the reading step, the presentation engine 6 of the reproducing apparatus 200 reads a markup document from the cache memory 3 in step 1301. In the loading step, the presentation engine 6 parses the markup document and generates a document tree in step 1302. If the markup document is not valid and a document tree is not generated in step 1303, an exception processing routine is performed in step 1304. If the markup document is valid and a document tree is normally generated in step 1303, the elements of the markup document are interpreted and formatting and rendering are performed in step 1305. Meanwhile, while the rendering is performed, event handlers for all kinds of events are enrolled in the script code interpreter 64. Event handlers listen whether an enrolled event is generated. If the markup document is rendered and corresponding AV data are decoded, the blender 7 blends the rendered markup document with decoded AV data streams, and outputs the result on the screen in step 1306. In the interacting step, the corresponding markup document is loaded on the screen, and the presentation engine 6 generates a "load" event to the script code interpreter 64 such that jobs to be performed in relation to the event can be processed. Then, interaction with the user is performed through the markup document in step 1307. Here, if there is a request to stop the presentation of the corresponding markup document in step 1308, the presentation engine 6 generates

an "unload" event to the script code interpreter 64 in step 1309. Then, in the finishing step, presentation of the current markup document is finished and presentation of the next markup document is prepared in step 1310. In the discarding step, the finished markup document is deleted from the cache memory 3 in step 1311. As described above, there may be a markup document in which the reading step follows immediately after the discarding step.

[Effect of the Invention]

According to the present invention as described above, when AV data are reproduced in the interactive mode, display compatibility is provided.

What is claimed is:

9. An apparatus for reproducing AV data including audio data and/or video data recorded on an information storage medium in interactive mode, the apparatus comprising:

a reader which reads and fetches data recorded on the information storage medium;

a cache memory which temporarily stores a markup document that is read by the reader; and

a presentation engine which presents the markup document according to a document life cycle which comprises a loading step for interpreting the markup document read by the reader and loading the document on a screen, an interacting step for performing interaction between the markup document loaded on the screen and the user, and a finishing step for finishing the presentation of the markup document.

10. The apparatus of claim 9, further comprising:

a buffer memory which buffers the AV data;

a decoder which decodes the AV data buffered in the buffer memory; and

a blender which blends the AV data decoded by the decoder and the markup document interpreted by the presentation engine, and outputs the blended result.

11. The apparatus of claim 9, wherein before the loading step the presentation engine performs a reading step for reading and fetching the markup document to the cache memory, as part of the document life cycle.

12. The apparatus of claim 9, wherein after the finishing step the presentation engine performs a discarding step for deleting the markup document remaining in the cache memory, as part of the document life cycle.

13. The apparatus of claim 9, wherein as the loading step, the presentation engine generates a document tree when the markup document is valid.

14. The apparatus of claim 9, wherein the presentation engine generates a document tree when the markup document is valid by checking DTD (document type definition).

15. The apparatus of claim 9, wherein the presentation engine performs rendering of a node of a document tree.

16. The apparatus of claim 9, wherein as the loading step, the presentation engine registers an event handler during the rendering of the markup document.

17. The apparatus of claim 16, wherein after the loading step, the presentation engine listens whether an event generates through the event handler.

18. The apparatus of claim 9, wherein as the loading step, the presentation engine performs the steps of:

(a) interpreting the markup document and generating a document tree; and

(c) based on the generated document tree, rendering the markup document.

19. The apparatus of claim 11, wherein the presentation engine further performs reading and fetching a stylesheet for the markup document to the memory.

20. The apparatus of claim 9, wherein the presentation engine further performs as the loading step:

(a) interpreting the markup document and generating a document tree;

(b) interpreting the stylesheet and applying the stylesheet to the document tree;

(c1) based on the stylesheet-applied document tree, generating a formatting structure; and

(c2) based on the generated formatting structure, rendering the markup document.

21. The apparatus of claim 20, wherein the presentation engine generates the document tree according to a rule that a root node of all nodes is set to a document node, a rule that all texts and elements generate nodes, and a rule that a processing instruction, a comment, and a document type generate a node.

22. The apparatus of claim 9, wherein in the loading step the presentation engine further performs generating a 'load' event.

23. The apparatus of claim 9, wherein in the interacting step when a finishing of a markup document loaded on a monitor is demanded, an 'unload' event is generated.

24. The apparatus of claim 9, wherein if an 'unload' event is generated in the interacting step, the presentation engine performs the finishing step.

25. An apparatus for reproducing AV data including audio data and/or video data recorded on an information storage medium in interactive mode, the apparatus comprising:

a reader which reads and fetches data recorded on the information storage medium;

a cache memory which temporarily stores a markup document and a stylesheet that are read by the reader; and

a presentation engine which comprises:

a markup document parser which interprets the markup document and generates a document tree,

a stylesheet parser which interprets the stylesheet and generates a style rule/selector list,

a script code interpreter which interprets a script code contained in the markup document,

a document object model (DOM) logic unit which modifies the document tree and the style rule/selector list according to interaction with the script code interpreter, and

a layout formatter/renderer which applies the document tree and stylesheet rule/selector list to the document tree, based on the applying, generates a formatting structure, and based on the generated formatting structure, renders the markup document.

26. The apparatus of claim 25, wherein the markup document parser generates the document tree according to a rule that a root node of all nodes is set to a document node, a rule that all texts and elements generate nodes, and a rule that a processing instruction, a comment, and a document type generate a node.

27. The apparatus of claim 25 or 26, wherein the presentation engine comprises a markup document step controller, and the markup document step controller generates a 'load' event to the script code interpreter if the rendering of the markup document is completed.

28. The apparatus of claim 27, wherein the step controller generates an 'unload' event to the script code interpreter in order to finish presentation of the markup document.

29 The apparatus of claim 25, further comprising:

a buffer memory which buffers the AV data;

a decoder which decodes the AV data buffered in the buffer memory; and

a blender which blends the AV data decoded by the decoder and the markup document interpreted by the presentation engine, and outputs the blended result.

5

FIG. 1

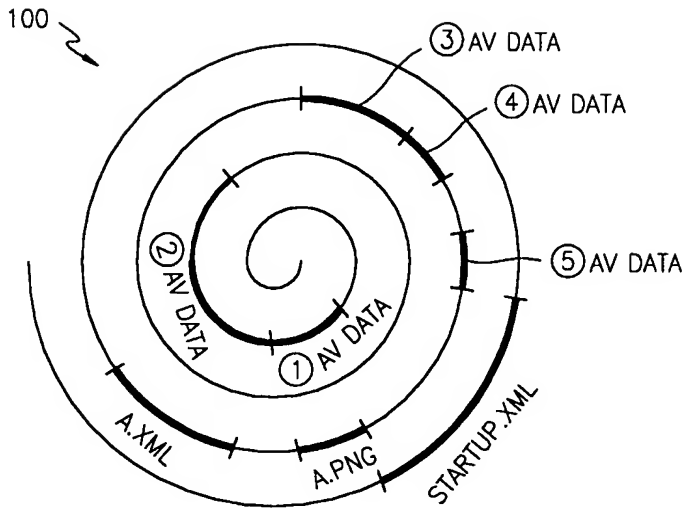


FIG. 2

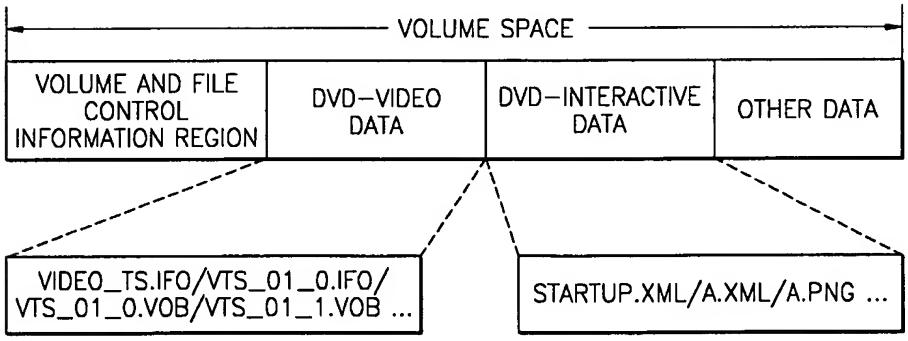


FIG. 3

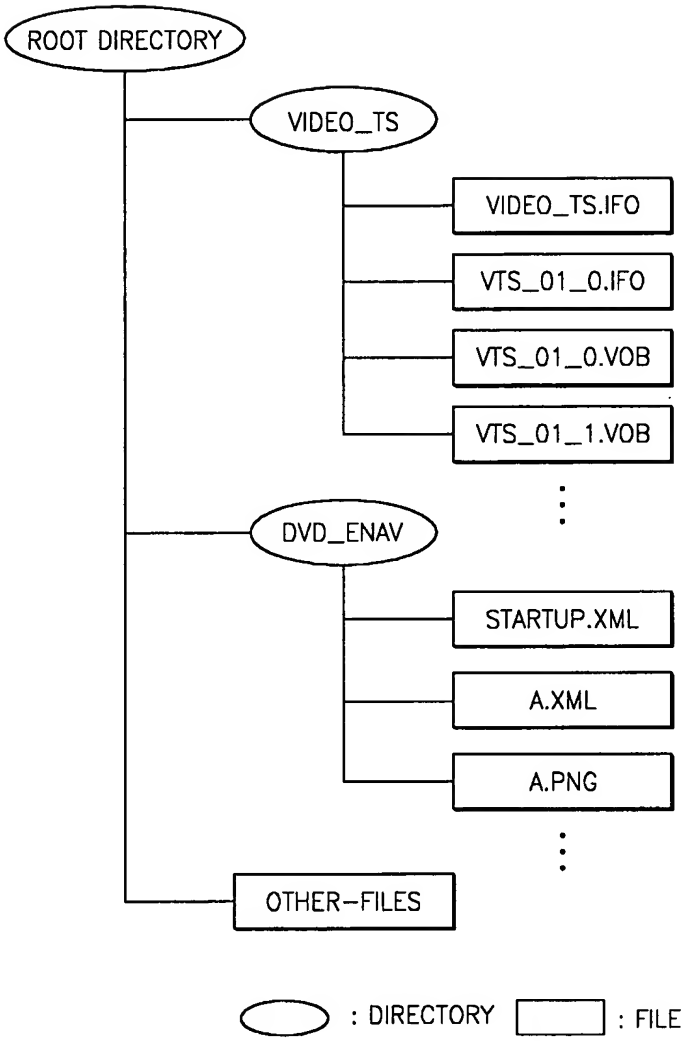

ROOT DIRECTORY
VIDEO_TS
VIDEO_TS.IFO
VTS_01_0.IFO
VTS_01_0.VOB
VTS_01_1.VOB
DVD_ENAV
STARTUP.XML
A.XML
A.PNG
OTHER-FILES
: DIRECTORY
: FILE

FIG. 4

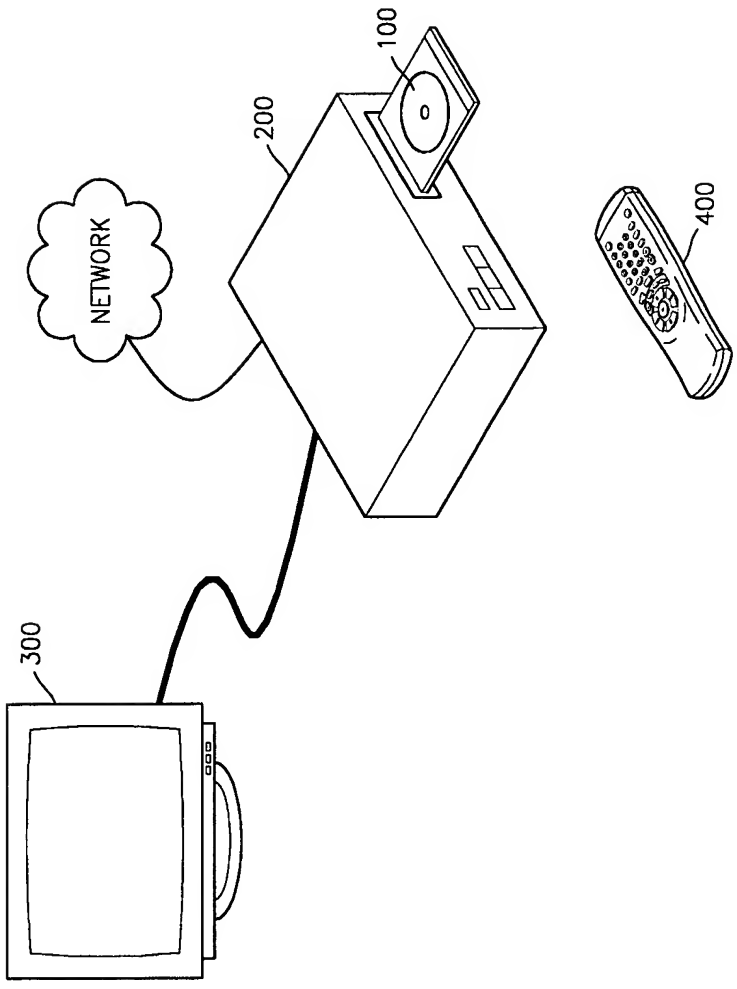


FIG. 5

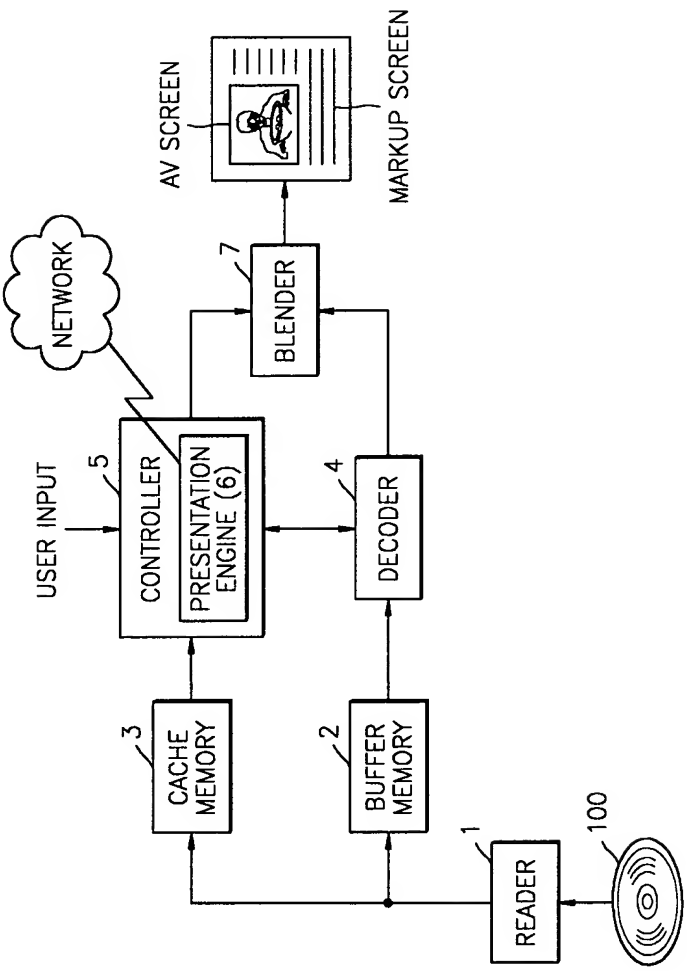

AV SCREEN
MARKUP SCREEN
NETWORK
BLENDER
7
USER INPUT
5
CONTROLLER
PRESENTATION ENGINE (6)
4
DECODER
3
CACHE MEMORY
2
BUFFER MEMORY
1
READER
100

FIG. 6

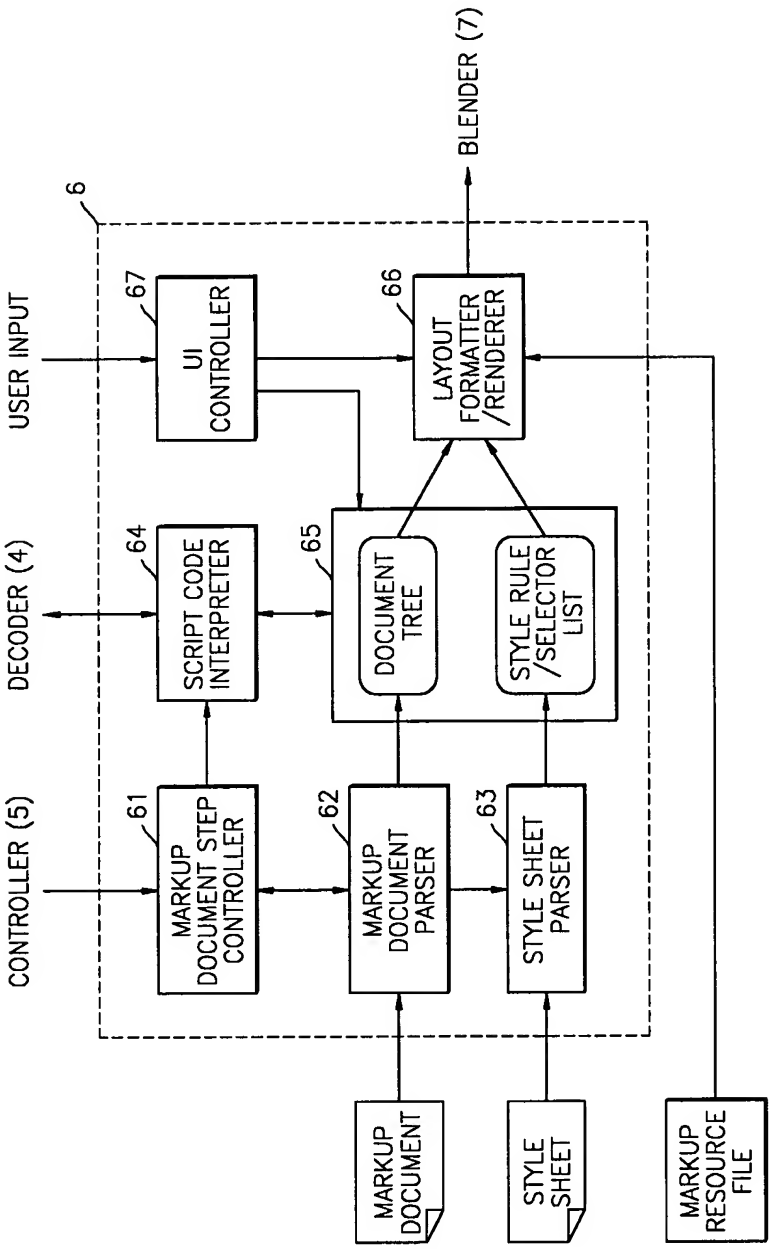

BLENDER (7)
6
USER INPUT
67
UI CONTROLLER
66
LAYOUT FORMATTER /RENDERER
DECODER (4)
64
SCRIPT CODE INTERPRETER
65
DOCUMENT TREE
STYLE RULE /SELECTOR LIST
CONTROLLER (5)
61
MARKUP DOCUMENT STEP CONTROLLER
62
MARKUP DOCUMENT PARSER
63
STYLE SHEET PARSER
MARKUP DOCUMENT
STYLE SHEET
MARKUP RESOURCE FILE

FIG. 7

```
<?xml version="1.0" ?>
<!—This is first comment -->
<!DOCTYPE html PUBLIC "-//DVD//DTD XHTML DVD-HTML 1.0//EN"
"http://www.dvdforum.org/enav/dtd/dvdhtml-1-0.dtd">
<html>
<head>
<title>DOM-core sample</title>
<script type="text/ecmascript">
alert("script code is in head tag")
function load_handler()
{
   alert("Hello, DVD-HTML")
   var strMsg="";
   strMsg+= document.documentElement.nodeName+"\n";
   strMsg+= document.documentElement.attributes.item(0).nodeName+"\n";
   var root= document.documentElement;
   for(var i=0;i<root.childNodes.length;i++)
   strMsg += "\t"+root.childNodes.item(i).nodeName+"\n";
   strMsg+= document.nodeName;
   alert(strMsg)
   alert(root.childNodes.item(0).childNodes.length);
alert(root.childNodes.item(0).childNodes.item(1).hasChildNodes());
}
function unload_handler()
{
   alert("good bye world")
}
</script>
</head>
<body onload="load_handler();" onunload="unload_handler();">
<script type="text/ecmacript">
alert(" before body text" )
</script>
<p>body text</p>
<script type="text/ecmascript">
<![CDATA[
alert("after body text")
]]>
</script>
</body>
</html>
```

FIG. 8

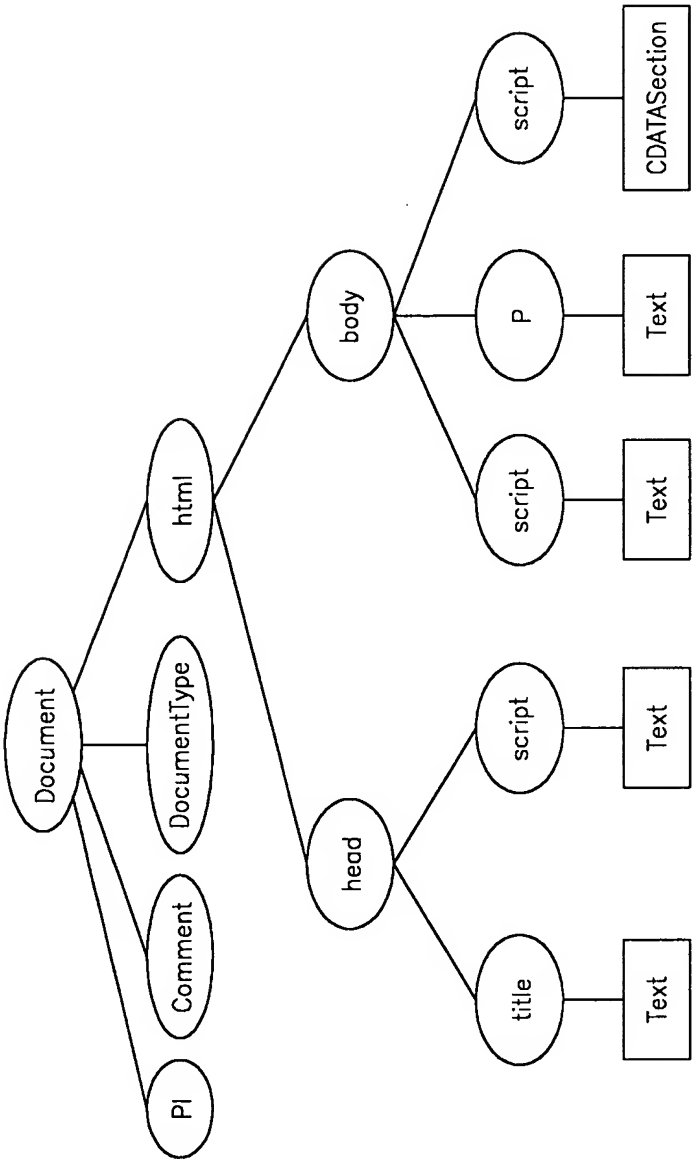

Document
PI
Comment
DocumentType
html
head
title
Text
script
Text
body
script
Text
P
Text
script
CDATASection

FIG. 9

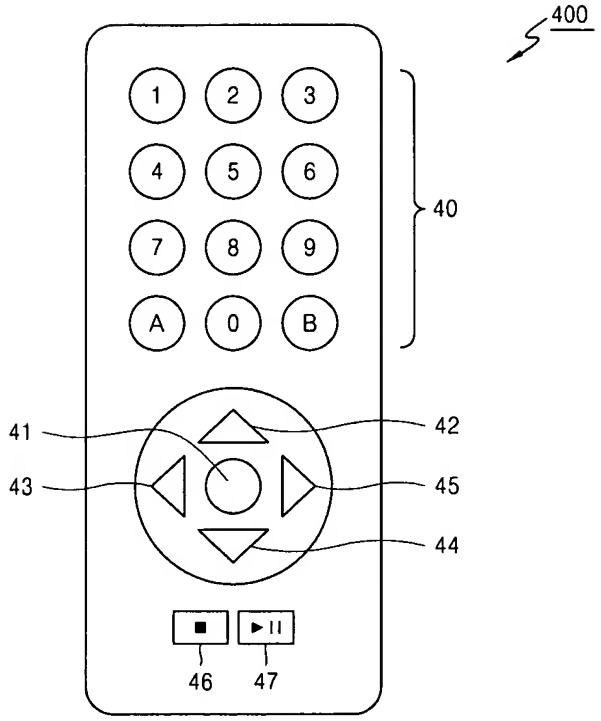

400
1
2
3
4
5
6
7
8
9
A
0
B
40
41
42
43
45
44
46
47

FIG. 10

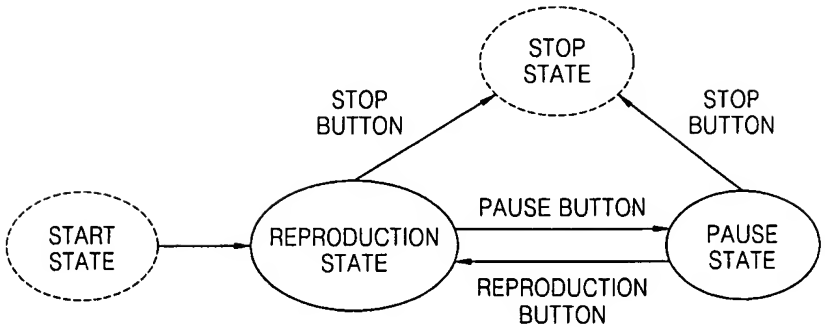

STOP STATE
STOP BUTTON
STOP BUTTON
START STATE
REPRODUCTION STATE
PAUSE BUTTON
PAUSE STATE
REPRODUCTION BUTTON

FIG. 11

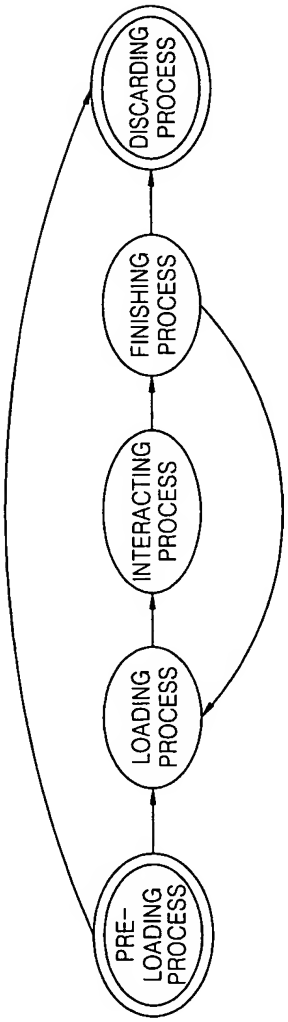

DISCARDING PROCESS
FINISHING PROCESS
INTERACTING PROCESS
LOADING PROCESS
PRE-LOADING PROCESS

FIG. 12B

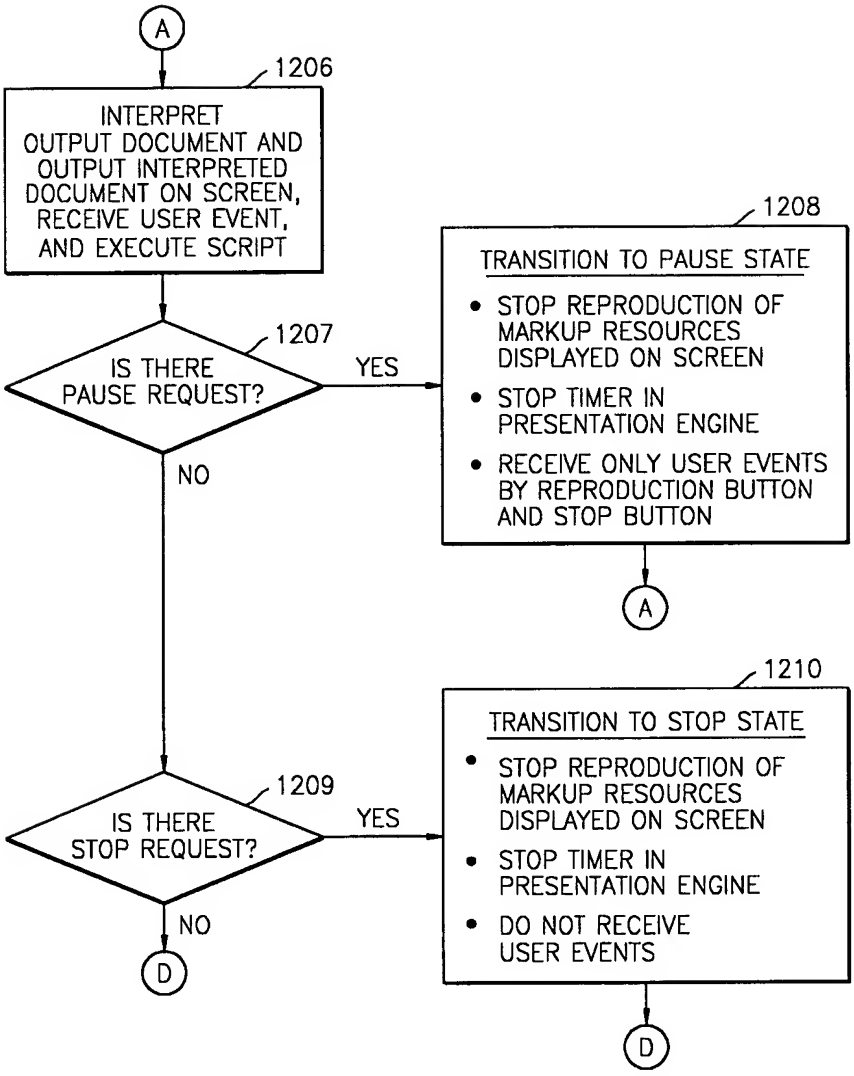

A
1206
INTERPRET OUTPUT DOCUMENT AND OUTPUT INTERPRETED DOCUMENT ON SCREEN, RECEIVE USER EVENT, AND EXECUTE SCRIPT
1207
IS THERE PAUSE REQUEST?
YES
NO
1208
TRANSITION TO PAUSE STATE
• STOP REPRODUCTION OF MARKUP RESOURCES DISPLAYED ON SCREEN
• STOP TIMER IN PRESENTATION ENGINE
• RECEIVE ONLY USER EVENTS BY REPRODUCTION BUTTON AND STOP BUTTON
A
1209
IS THERE STOP REQUEST?
YES
NO
D
1210
TRANSITION TO STOP STATE
• STOP REPRODUCTION OF MARKUP RESOURCES DISPLAYED ON SCREEN
• STOP TIMER IN PRESENTATION ENGINE
• DO NOT RECEIVE USER EVENTS
D

FIG. 12C

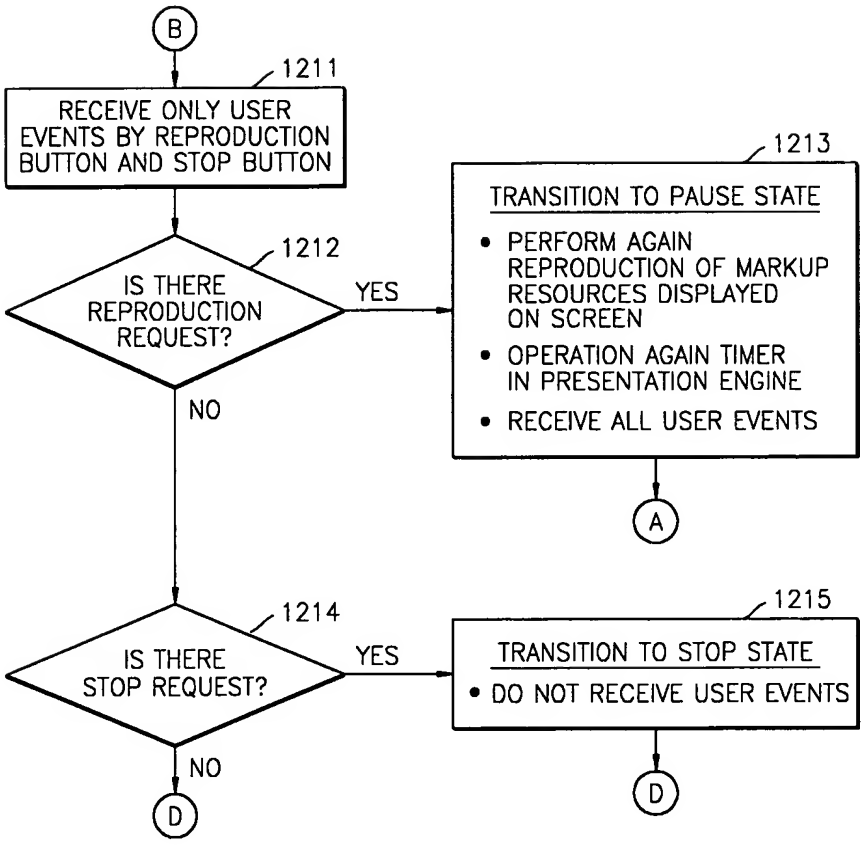

B
1211
RECEIVE ONLY USER EVENTS BY REPRODUCTION BUTTON AND STOP BUTTON
1212
IS THERE REPRODUCTION REQUEST?
YES
NO
1213
TRANSITION TO PAUSE STATE
• PERFORM AGAIN REPRODUCTION OF MARKUP RESOURCES DISPLAYED ON SCREEN
• OPERATION AGAIN TIMER IN PRESENTATION ENGINE
• RECEIVE ALL USER EVENTS
A
1214
IS THERE STOP REQUEST?
YES
NO
1215
TRANSITION TO STOP STATE
• DO NOT RECEIVE USER EVENTS
D
D

FIG. 12D

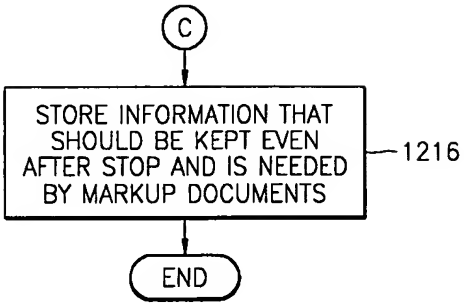
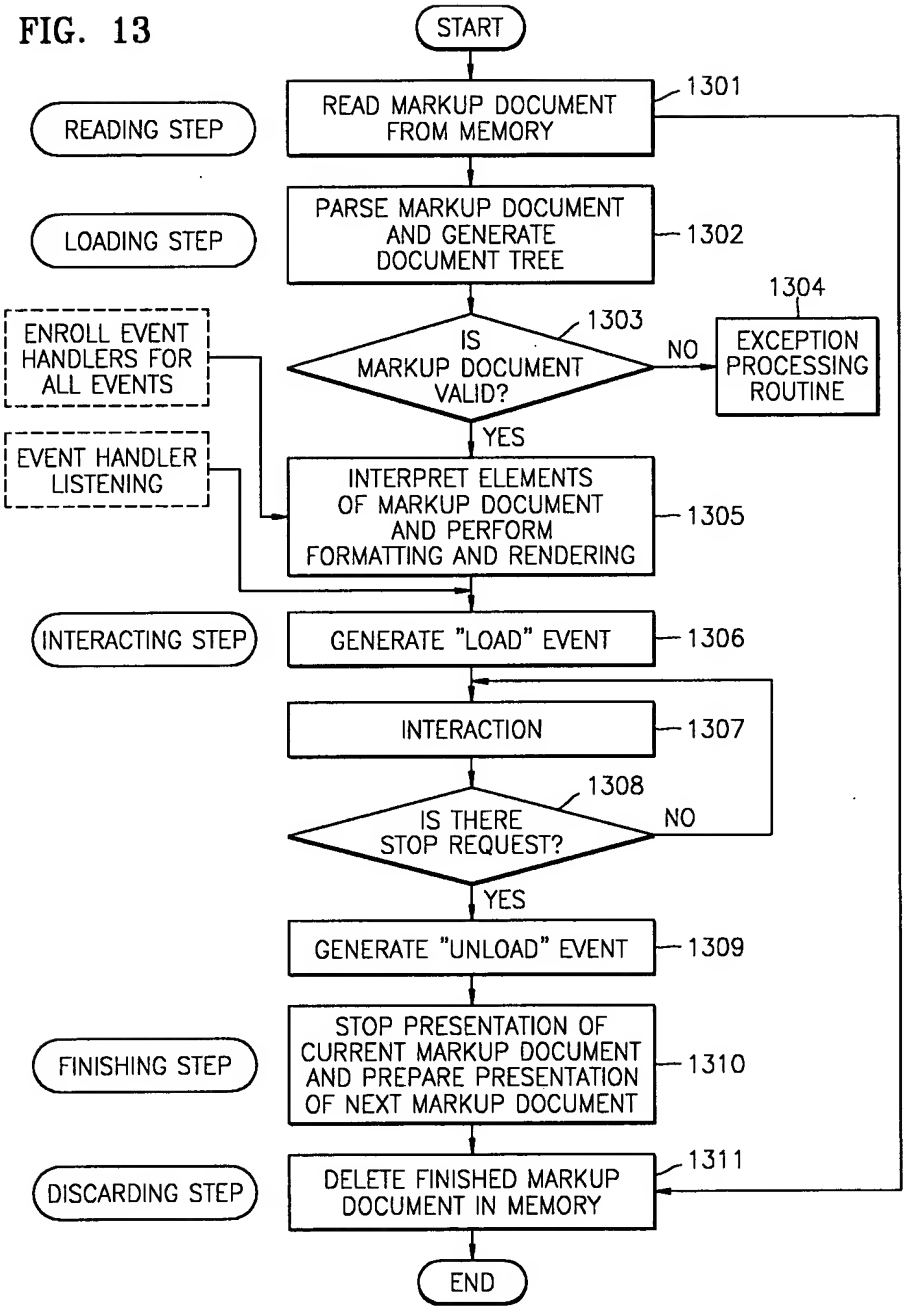

C
STORE INFORMATION THAT SHOULD BE KEPT EVEN AFTER STOP AND IS NEEDED BY MARKUP DOCUMENTS
1216
END

FIG. 13


START
READING STEP
READ MARKUP DOCUMENT FROM MEMORY
1301
LOADING STEP
PARSE MARKUP DOCUMENT AND GENERATE DOCUMENT TREE
1302
1304
ENROLL EVENT HANDLERS FOR ALL EVENTS
IS MARKUP DOCUMENT VALID?
1303
NO
EXCEPTION PROCESSING ROUTINE
YES
EVENT HANDLER LISTENING
INTERPRET ELEMENTS OF MARKUP DOCUMENT AND PERFORM FORMATTING AND RENDERING
1305
INTERACTING STEP
GENERATE "LOAD" EVENT
1306
INTERACTION
1307
IS THERE STOP REQUEST?
1308
NO
YES
GENERATE "UNLOAD" EVENT
1309
FINISHING STEP
STOP PRESENTATION OF CURRENT MARKUP DOCUMENT AND PREPARE PRESENTATION OF NEXT MARKUP DOCUMENT
1310
DISCARDING STEP
DELETE FINISHED MARKUP DOCUMENT IN MEMORY
1311
END